

Real-Time Programming

Summer Semester 08

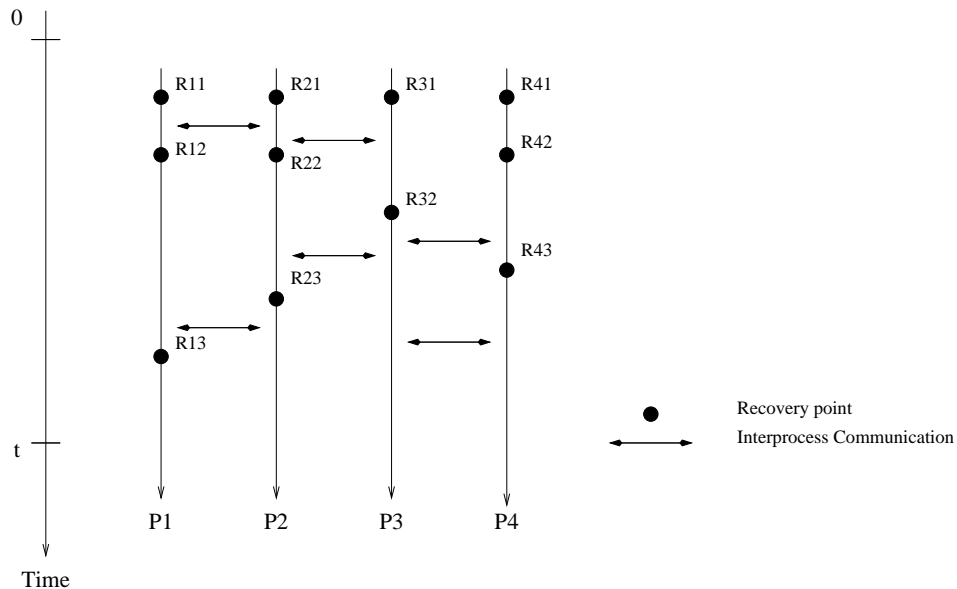
Assignment #3

Homepage: http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/

Discussion of Solutions: June 4, 2008

1 Recovery Points

The following diagram illustrates the concurrent execution of four communicating processes and their associated recovery points (for example, R11 is the first recovery point for process P1).



Explain what happens when an error is detected by P1 at time t . Do the same for P2, P3, and P4.

2 Predefined Exceptions in Ada

The following program `buggy.adb` given at

http://www.iste.uni-stuttgart.de/ps/Lehre/SS2008/V_RTP/a3/buggy.adb

contains errors that cause constraint violations during runtime. Such constraint violations cause `Constraint_Error` exceptions. Locate the errors (without executing the program).

```

with Ada.Numerics.Discrete_Random;
with Ada.Text_IO;
with Unchecked_Deallocation;

procedure Buggy is

  type Priority_Range is new Positive range 1 .. 5;

  type State is (Ready, Waiting, Idle, Zombie);

  type Parameter_Less_Procedure is access procedure;

  procedure Do_Something is
  begin
    Ada.Text_IO.Put_Line ("I am doing something...");
  end;

  type Task_Descriptor is record
    Priority      : Priority_Range := Priority_Range'Last;
    Current_State : State := Ready;
    Code         : Parameter_Less_Procedure := Do_Something'Access;
  end record;

  type Task_Descriptor_Access is access Task_Descriptor;

  procedure Free is new Unchecked_Deallocation
    (Task_Descriptor,
     Task_Descriptor_Access);

  type Index is new Positive;

  package Random_Index is new Ada.Numerics.Discrete_Random (Index);

  type Waiting_Queue is array (Index range <>) of Task_Descriptor_Access;

  function Highest_Priority_Task (Q : Waiting_Queue)
  return Index
  is
    Max_Ind : Index := 1;
  begin
    for I in Index (2)..Index (100) loop
      if Q(Max_Ind).Priority < Q(I).Priority then
        Max_Ind := I;
      end if;
    end loop;
    return Max_Ind;
  end Highest_Priority_Task;

```

```

procedure Execute_Next (Q : in out Waiting_Queue) is
  Next : Index := Highest_Priority_Task (Q);
begin
  Ada.Text_IO.Put (Index'Image (Next) & ": ");
  Q (Next).Code.all;
  Q (Next).Priority := Q (Next).Priority - 1;
end;

procedure Kill (A_Task : in out Task_Descriptor) is
begin
  A_Task.Current_State := Zombie;
  A_Task.Priority      := Priority_Range'First;
  A_Task.Code          := null;
end Kill;

procedure Generate (Q : in out Waiting_Queue) is
begin
  for I in Q'Range loop
    Q (I) := new Task_Descriptor;
  end loop;
end Generate;

procedure Clean_Up (Q : in out Waiting_Queue) is
begin
  for I in Q'Range loop
    if Q (I).Current_State = Zombie then
      Free (Q(I));
    end if;
  end loop;
end Clean_Up;

Queue          : Waiting_Queue (1 .. 100);
Random_Generator : Random_Index.Generator;
begin
  Generate (Queue);
  loop
    Execute_Next (Queue);
    if Random_Index.Random (Random_Generator) < 10 then
      Kill (Queue (Random_Index.Random (Random_Generator)).all);
    end if;

    if Random_Index.Random (Random_Generator) < 3 then
      Clean_Up (Queue);
    end if;
  end loop;
end Buggy;

```